Quality-driven Automatic Transformation of Object-Oriented Navigational Models*

Cristina Cachero[1], Marcela Genero[2], Coral Calero[2], and Santiago Meliá[1]

[1] Web Engineering and Databases Research Group
University of Alicante. Spain
{ccachero,smelia}@dlsi.ua.es
[2] ALARCOS Research Group
University of Castilla La Mancha. Spain
{Marcela.Genero,Coral.Calero}@uclm.es

Abstract. Navigability is a main concern in the design of Web applications. In order to assess such navigability a number of measures has been proposed. From them, measures defined on conceptual models are specially relevant, as it is well known that high quality conceptual models are critical to the success of the deployed system. However, measurement methods associated to such measures, as well as the design modifications that need to be performed on the models in order to improve their values, are usually tightly coupled with particular Web Engineering approaches. This fact compromises their effectiveness and their propagation capacity to different environments and/or methodologies.
Our aim in this paper is to illustrate how navigability measures can be captured in a general manner, by instantiation of a measuring metamodel that is based on the Software Measurement Ontology proposed by García *et al.* In this way, not only is it possible to define a reusable set of relevant measures for a given family of applications, but also such measures can be queried in the context of MDA transformation rules. These rules capture both the measure decision criteria and the design modifications that should take place if the measure value for a given navigational model does not match such criteria.

1 Introduction

The ever increasing complexity of Web applications has caused the Web Engineering field, defined as the discipline that is concerned with the application of systematic and quantifiable approaches to the cost-effective development and evolution of high-quality applications in the World Wide Web [18], to evolve at

* This paper has been supported by the Spain Ministry of Science and Technology, project numbers TIN2004-00779, TIN2005-25866-E and TIC2003-07804-C05-03. Also, this research is part of the DADASMECA project (GV05/220), financed by the Valencia Government and the DADS (PBC-05-012-2) and the DIMENSIONS (PBC-05-012-1) projects, financed by the Regional Science and Technology Ministry of Castilla-La Mancha (Spain).

an extremely fast pace. This discipline intertwines sound Software Engineering principles with a suitable set of abstractions, particular to the idiosyncrasy of the Web. However, and despite its definition, the inclusion in the different Web Engineering proposals of guiding principles that contribute to guarantee that the resulting applications comply with a set of desirable quality characteristics is still a challenge for the field. In fact, up to now no significant relationship has been empirically demonstrated between the quality of the Web application and the fact that the designer has followed or not a given Web methodology.

This situation undoubtedly contributes to the fact that, still nowadays, the use of 'creative' approaches for the development of Web applications is the commonality. Such creative approaches assume that the designer is a navigability expert, and therefore skillful in the use of a set of common-sense practices that, usually aimed at improving the interface usability, have proven successful in a bunch of well known applications [23], and therefore assumed to be universally right, even if most of them (over 60%) lack empirical validation [13]. With this kind of approaches, quality assessment must be performed over the deployed products, often with the aid of Web-devoted inspection techniques such as the Systematic Usability Evaluation (SUE) method [5] or any of the myriad of automated html-content validation tools (e.g. [1, 21], to name a few).

Although assessing quality at such a late stage of development can be useful, it is commonly avowed that early detection of problems in the artifacts produced in the initial phases of the life-cycle can save time, cut production costs, and raise the final desired product quality [11]. Aware of such potential benefits, the Web Engineering community has recently started to invest efforts in the inclusion of early measures to guide the construction of the different Web models, with special emphasis on domain and navigational models.

From them, domain models for Web applications hoard the highest number of early measure proposals [8]. One reason for this fact may be that most of the measures proposed for UML class diagrams can be seamlessly tailored to Web domain models [16]. Regarding navigational models, which are a distinctive feature of Web Engineering proposals, most efforts have been directed towards the promotion of a good use of the conceptual navigation constructs provided by each approach. For example, WebML defines a WebML Quality Analyzer [14] that is able to automatically check the XML specification of WebML conceptual schemas, and verify and measure some internal attributes, such as the consistency or completeness of the models. Similarly, UWE [7] pays special attention to usability, and recommends the integration of guidelines into both the design process and the CAWE tools that usually accompany Web proposals. Going one step further, the work of Abrahao *et al.* [2] proposes a set of measures that capture some available heuristics [19].

In fact, navigability is at the core of quality aspects such as usability or maintainability [2, 23, 3, 4], as it is widely recognized that a good navigation design let users acquire the information they are seeking quickly and efficiently.

However, much work remains to be done. To our knowledge, all the proposed measures for Web models are tightly coupled with the Web Engineering approach

for which they were defined. Unfortunately Web approaches usually greatly differ not only in notation but also in the semantics associated to the constructs, hampering the measure propagation to different environments and/or methodologies. Also, measures are usually defined and calculated in an ad-hoc manner, and the effects of the measure values on the models (that is, the actions the designer could perform in order to improve the model) are usually not specified.

Therefore, our aim in this paper is threefold. On one hand, we aim at demonstrating how the definition of a navigational meta-model not only facilitates the understanding of the different Web approaches but also serves as a basis on which early measures for navigational models can be defined and calculated. For this purpose, in Section 2 we present a partial view of the OO-H navigational meta-model, as well as how navigability measures can be formally expressed over this meta-model by means of OCL expressions [28]. On the other hand, we claim that the Web Engineering field can benefit from defining loosely coupled Web measuring models, that is, models that are applicable to a whole family of Web applications regardless of the chosen Web approach. Fortunately, we have at our disposal a Software Measurement Ontology (SMO) [15]. and its corresponding meta-model that allows to derive concrete measurement models. Our proposal in this sense is to instantiate a subset of this meta-model to express in an independent way a set of navigability measures. In Section 3 we illustrate this approach, and provide as an example a meta-model instantiation that reflects the navigability measure presented in Section 2. Last, we need a way to automate the evaluation/evolution of the navigational model depending on the chosen measures. For this purpose in Section 4 we use a QVT transformation rule [24] that, departing from the OO-H navigational meta-model and the SMO meta-model, generates a new OO-H navigational model that is checked against the quality decision criteria, which are also expressed in the meta-model instantiation. Section 5 concludes this paper with a summary of the main contributions and some future lines of research.

2 The OO-H Navigation Model

As we have stated above, it is well known that the improvement of the early artifacts produced in a development process has a great impact on the final product cost and quality [11]. Also, it is commonly avowed that, when developing Web applications, one of the most important early artifacts that must be produced is the navigational model. The Navigational model reflects the paths the user must follow through the information domain in order to achieve her goals. It is usually organized around navigational packages (also named contexts or targets in different approaches), each one encapsulating views defined over domain objects and paths to connect them. The names of the constructs, as well as their corresponding decorators, usually diverge from proposal to proposal, reflecting their different roots and degree of evolution. Due to this lack of a common Web conceptual ontology, it is highly recommendable for proposals to provide a navigational meta-model that systematically and unambiguously

defines the concepts involved and their relationships, and facilitates in this way the communication among researchers and/or practitioners.

Such has been the approach taken by OO-H [17], a well known Web Engineering method whose navigational meta-model is partially presented in Fig. 1. Due to space reasons, we have intentionally left out details concerning service invocation, as well as the existing relationships with UML meta-classes.

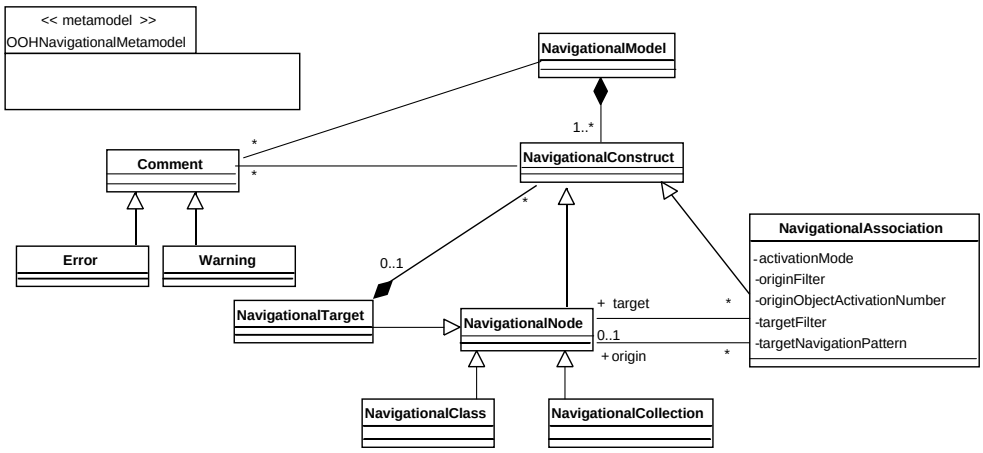

Fig. 1. *Partial View of the OO-H navigational meta-model*

In Fig. 1 we can observe how, as in most Web Engineering approaches, the main constructs of the OO-H Navigational Model are *Navigational Targets* (NT), *Navigational Classes* (NC), *Navigational Associations* (NA) and *Navigational Collections* (C). A Navigational Target is a packaging mechanism that serves to structure the navigation through the application subsystems. Navigational Classes are views over conceptual classes, and reflect the information that makes up the current view together with the operations that can be invoked from each view. Navigational Associations reflect navigation steps through the information. This is the richest construct in OO-H, which in this sense greatly differs from other approaches where the main construct is the Navigational Class. The attributes of the association meta-class allow for the specification of not only the population of the target view (*targetFilter*) and the navigation structure through this population (*targetNavigationPattern*) but also the objects from which such navigation is possible (*originFilter*), the cardinality of the origin set of objects (*originObjectNavigationNumber*) and whether the user interacts or not with the application in order to activate such navigation (*activationMode*). Last, Navigational Collections are access mechanisms (menus) that group together navigation paths.

To illustrate their use, let's imagine that we want to model a Ticket Sales system. A Navigation Model example corresponding to this system is presented in Fig. 2.

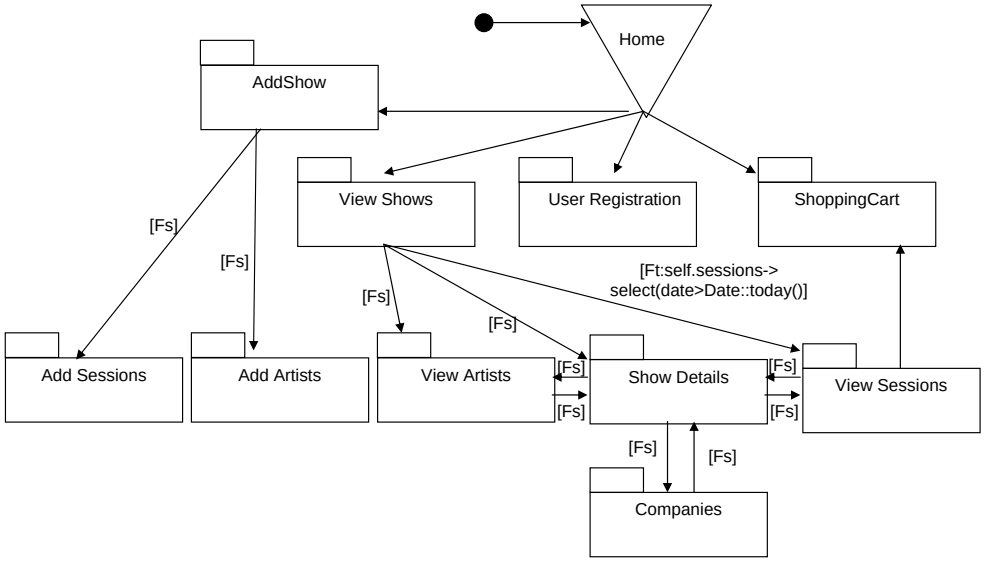

Fig. 2. *OO-H Navigational Model corresponding to a Ticket Sales system*

The partial navigation model of Fig. 2 reflects a system whose navigation is organized around many NT (e.g. *User Registration*, *Shopping Cart*, *Show Details*, and so on), each one encapsulating the navigation paths needed to fulfill a subset of system requirements. The different NT are accessed via a *Home* Collection that, depicted as an inverted triangle, represents the application main menu. The NT and the Home collection are connected by means of NA which, as stated above, may have one or more *Filters* associated. Filters are OCL expressions [26] that constrain navigation. In OO-H, filters can be either user-defined or automatically generated, based on the domain conceptual associations. In our example, the NA between *View Shows* and *View Artists* is an example of NA adorned with a predefined (structural) filter (Fs): the filter extracted from the relationship between shows and their corresponding artists. On the other hand, the filter *self.sessions->select(date>= Date::today())* that adorns the NA between *View Shows* and *View Sessions* is an example of target filter (Ft) that restricts the set of target objects to those sessions that have not yet taken place. In fact Fig. 2 represents one of the most evident mistakes novice web designers make when first designing Web applications using OO-H: they tend to define a new NT for each requirement, instead of performing a previous grouping task. This causes a poorly structured navigational tree and augments the model complexity. In fact, the impact of the number of NT on the model complexity has already been

assessed in [2], through the Number of Navigational Contexts measure. For the sake of coherence with OO-H, in the remaining of the paper we have renamed this measure *Number of Navigational Targets* (NNT).

Up to now, this measure could be defined in the context of the VisualWADE tool [27], the Computer Aided Web Engineering (CAWE) tool that supports OO-H, as follows:

```
context NavigationalModel
def NNT:Integer=self.navigationalConstruct->select(oclIsTypeOf(NavigationalTarget))->size()
```

We would like to emphasize how the use of OCL, as stated in [9, 10, 25], allows for a formal and unambiguous measure definition, and improves its understandability.

In VisualWADE it is also possible to define, using OCL, restrictions over these measures. At this point we face a problem: if little work has been done on validating the impact of measures on navigability, much less has been done on assessing threshold values for such measures. However, for the sake of the example, let's assume that, for this kind of application, the famous Miller's 7+-2 rule [22] is applicable, that is, we do want a navigational model that neither is too scattered (more than nine NT) nor too compressed (less than four NT). This restriction may be expressed as follows:

```
context NavigationalModel
inv notTooManyNT: NNT<10
inv notTooFewNT: NNT>4
```

If we look back to Fig. 2 we can observe that the `notTooManyNT` invariant is not fulfilled, what in VisualWADE would cause a warning to be raised. Although this approach works well for prototypical development, it has many drawbacks. On the one hand, OCL rules associated with the OO-H meta-model may need to be changed whenever we change the kind of application we are modelling (as measures that are relevant for a certain family of applications may not be applicable to others). On the other hand, this way of defining quality measures is highly coupled with the particular OO-H constructs and their corresponding semantics, what causes that practitioners familiarized with other methodologies may find it difficult to understand and apply the measures to their own models. Therefore we propose to go one step further, and provide a way to define measures that is independent from the chosen methodology and its corresponding meta-model, facilitating reusability.

Next we explain how we can achieve this goal.

3 Adaptation of the SMO to navigational models

The homogenization and systematization of the concepts that are relevant for a given domain is a problem that has gained popularity since the advent of the Semantic Web. This homogenization has been achieved through a number of ontologies, defined as formal explicit specifications of a shared conceptualization.

The definition of measures using a standard ontology improves the understandability and communication of software measures, as it establishes a common vocabulary for developers [6]. For all these reasons, we have decided to adhere to the SMO presented in [15] (see Fig. 3) for the description of the navigational measures that are applicable to a given family of Web applications. The SMO

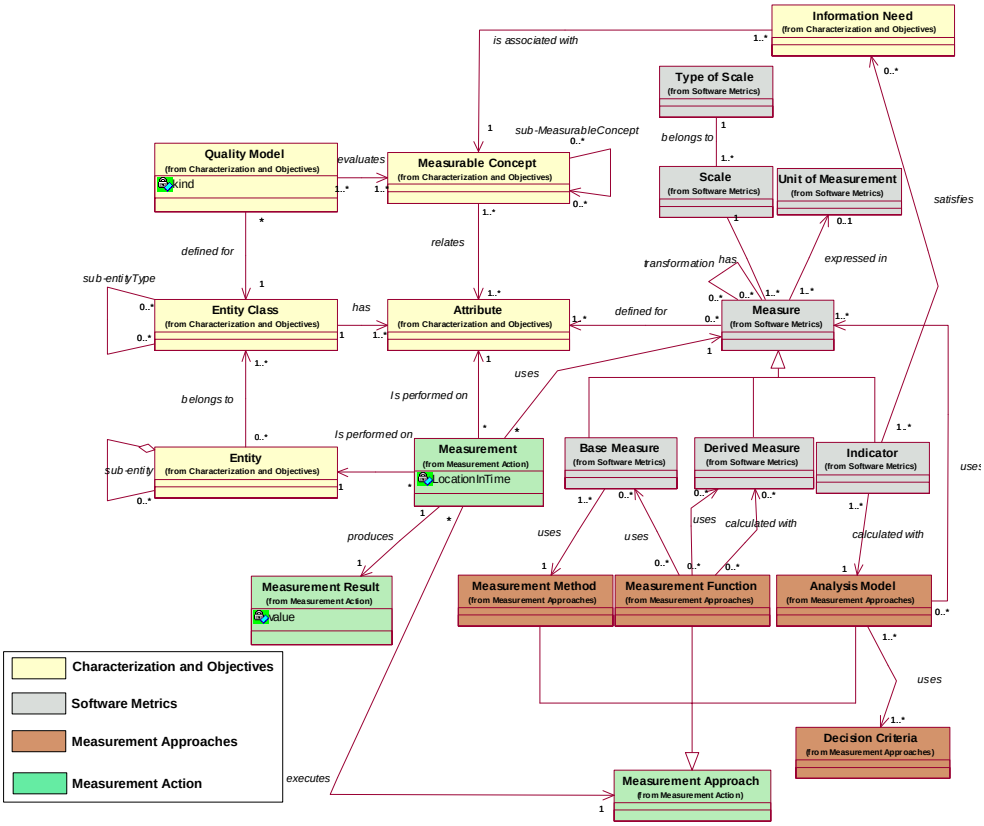

Fig. 3. *UML Class Diagram of the Software Measurement Ontology (SMO)*

ontology is divided into four sub-ontologies, as we can observe in Fig. 3. Such sub-ontologies are: (1) the way of measuring (*MeasurementApproaches*), (2) the action of measuring (*Measurement*), (3) the result obtained (*Software Measures*) and (4) the concept and context of measure (*Characterization and Objectives*). Each sub-ontology provides a set of concepts, some of them optional, that may contribute to the definition of the measure.

This ontology contributes to the definition of the measure. Due to the relevance of deriving concrete measurement models, a meta-model has been defined based on SMO. This meta-model is divided in four packages, one related to each

sub-ontology. The fact that our measuring model is based on an ontology-aware meta-model makes it shareable among Web proposals [6]. Next we present the way in which we can instantiate the SMO meta-model to reflect our NNT measure example.

3.1 Ontology-aware measuring meta-model Instantiation

As the reader may have already inferred, the definition of the NNT measure (see Fig. 4) corresponds to the need to *assess the navigability of the Navigational Model*. In order to fulfill this **Information Need**, our measure is part of a *OO-H* **Quality Model** that gathers all the measures, decision criteria etc. that are applicable to any OO-H entity. This model is aimed at evaluating the *navigability* **Measurable Concept**. The *OO-H* **Quality Model** is made up of a set of **Attributes**, among which the *Navigational Complexity* is the one related with the NNT measure. All these concepts correspond to the Characterization and Objectives package that reflects the namesake sub-ontology that we presented in Fig. 3.

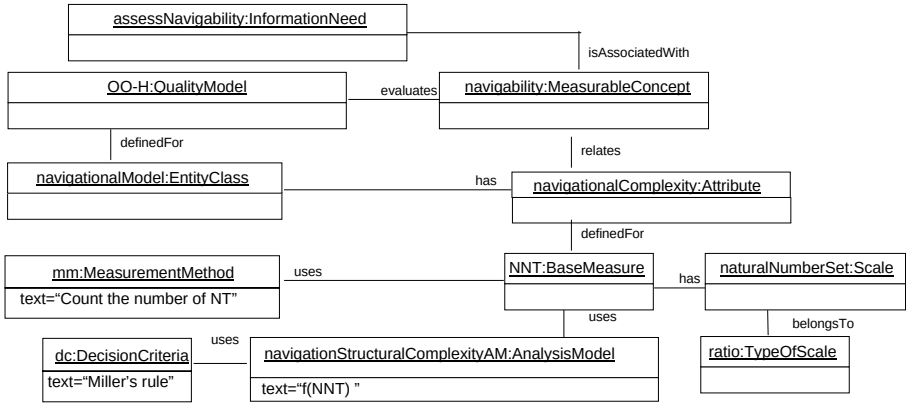

Fig. 4. *Ontology-aware measuring model that reflects the NNT measure*

If we now move on to the Software Metrics package (see Fig. 3), we find other relevant concepts for our purpose; namely, we need to express that the afore mentioned *Navigational Complexity* attribute is going to be measured, among others, with the aid of the *NNT* **Base measure**. The fact that the measure is of subtype Base implies that it does not depend on any other measure to calculate its value. Also, we want to express that the value of the NNT measure belongs to a *natural number* **Scale**, of *ratio* **Type of scale**. The **Measurement Unit** would be the *Navigational Target*.

The only thing left for the completion of the NNT measure definition is the specification of how we intend to measure such concept. We can do so by

instantiating the Measurement Approaches package related with the namesake sub-ontology (see Fig. 3). This package provides the necessary meta-classes to express that the **Measurement Method** used to calculate the NNT measure consists on counting the number of NT that a given navigational view includes.

Also, we need to specify the **Analysis Model**, that includes a set of **Decision Criteria**. In our example, as we have just defined one measure, our Analysis Model assesses navigability just depending on the value of the NNT measure. The Decision Criteria on its turn establishes that a good navigability value has to comply with the Miller's rule, what implies that, in order for the model to be not too trivial nor too complex, the NNT must be between five and nine.
Once the NNT measure has been defined, it is time to see how we can apply it to a given navigational model, such as the one presented in Fig. 2.

4 Automation of Measures

One of the main advantages of the measuring model presented in Fig. 4 is its capacity of reuse, as it does not assume any particular Web Engineering navigational meta-model. However, in order to be able to apply a given measure to a concrete navigational model, such connection needs to be established. This can be easily done if we regard a navigational model as a subtype of the concept **Entity Class**. This connection opens the path to the application of a Model-Driven Engineering approach [20] to automate the navigability assessment.

Back to our example, in Fig. 2 we can intuitionally observe how the OO-H navigational model does not fulfill the rule of 7+-2 navigational targets. Therefore an expert designer would manually restructure the application to fulfill such rule. The automation of this process can be achieved by means of a set of transformation rules that, expressed in QVT [24], allows to encapsulate all the knowledge particular to a given Web Engineering approach (in our case OO-H).

Let's illustrate this approach by depicting a possible transformation rule that counts the number of NT for the navigational model of Fig. 2 and annotates the model if the decision criteria is not fulfilled (see Fig. 5).

In Fig. 5 the QVT graphical notation for the *NumberOfNavigationalTargets* relation is presented. This transformation rule involves two checkonly (c) domains: the *NavigationalModel* domain (root for the OO-H meta-model) and the *BaseMeasure* domain (defined in the context of the measuring meta-model). First, the transformation rule checks whether the NNT measure is relevant for our navigational model, and whether Miller's decision criteria is applicable. This will be evaluated to true if the corresponding objects are present in the measurement model (meta-model instantiation) that we have previously defined (see Fig. 4).

Then we must calculate the actual number of NT included in the navigational model under consideration. In OO-H this value can be established by simply counting the number of NT associated to our Navigational Model. The transformation rule stores this value in the *nts* variable.

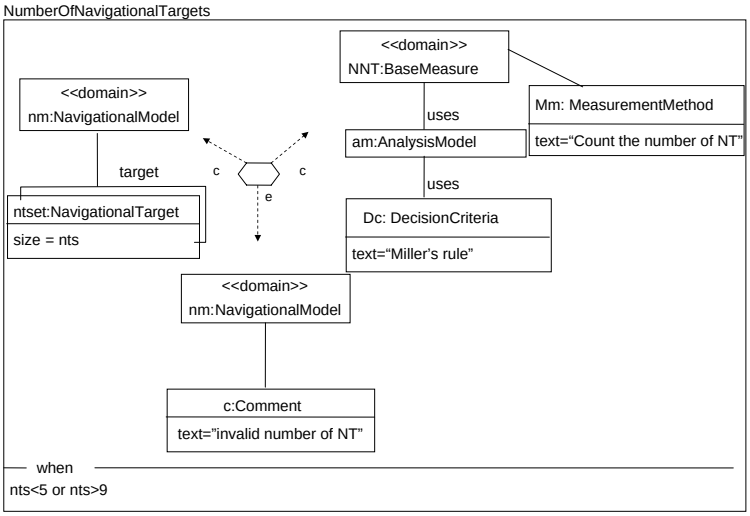

Fig. 5. *QVT Transformation rule that checks OO-H navigational models for Miller's rule violations*

The adaptation of Miller's rule to the OO-H meta-model is established in the *when* clause of the transformation. This clause indicates, also in OCL-like syntax, that the transformation rule must be activated only if the number of NT is erroneous. In this case, the desired action is to enrich the OO-H model with a comment, associated with the whole model, that warns the designer about the violation of the rule. We specify such action on a third enforceable (e) domain (again the OO-H *NavigationalModel*). More transformation rules can be defined in this general way, making up a repository of measuring transformation rules.

We would like to stress the fact that the decision whether or not a particular rule is relevant for a given application will be taken once the measuring meta-model has been instantiated for such application. If the measuring structure reflected in the transformation rule is present in the meta-model instantiation, the rule will check whether the decision criteria is met, and will take any desired action if this is not the case. Although in our example the action has consisted on simply annotating the model, more sophisticated transformation rules could be defined to automatically generate a new model that does comply with the measuring criteria.

5 Conclusions and further work

This paper has presented a way to define a reusable measuring model that, based on the SMO meta-model, can be integrated into any particular Web Engineering approach. Also, it has demonstrated how the instantiation of this meta-model can participate in the quality assessment of particular navigational models, becoming a discriminator to decide whether or not a given QVT transformation

rule, defined as part of a transformation rule repository, is applicable. Such transformation rules are the only elements that are aware of the specific Web Engineering meta-model that is being used, encapsulating in this way the specific knowledge. Besides, these transformation rules automate the measurement process, as well as the annotation/modification of the corresponding navigational models (if necessary) depending on the established measuring decision criteria. We would also like to stress the fact that the explicit consideration of ontologies (SMO) and standards (UML, OCL, QVT) whenever possible improves understandability and reusability of the approach.

At this moment efforts are being made towards the definition of transformation rules that not only annotate but also modify in a sound way the OO-H navigational models. Also, intensive work is being performed on the OO-H CAWE tool to provide full support to this proposal.

We are aware that a lot of work is left to define and empirically validate relationships between measures, measurement models and specific design modifications. For example, the assumption that the ideal number of NT for a medium-size e-commerce application such as our Ticket Sales system follows Miller's rule has not, to our knowledge, been either refuted nor confirmed yet.

Last, we would like to stress how, as soon as the Web Engineering community reaches an agreement regarding a common meta-model for Web application development, not only the meta-model instantiation but also the set of defined transformation rules will be able to be seamlessly reused among approaches.

References

[1] Web accessibility verifier. http://aprompt.snow.utoronto.ca.

[2] S. Abrahao, N. Condory-Fernandez, L. Olsina, and O. Pastor. A Defining and Validating Metrics for Navigation Models. In *Proceedings of the 9th International Software Metrics Symposium*, pages 200–210, 2003.

[3] J. Almer. Designing for Web Site Usability. *IEEE Computer*, 35(7):102–103, 2002.

[4] J. Almer. Web Site Usability, Design and Performance Metrics. *Information Systems Research*, 13(2):151–167, 06 2002.

[5] A. De Angeli, M. Matera, M.F. Costabile, F. Garzotto, and P. Paolini. Validating the SUE inspection technique. In *Proceedings of the working conference on Advanced Visual Interfaces*. ACM Press, 2000.

[6] U. Assmann, S. Zschaler, and G. Wagner. *Ontologies in Software Engineering and Software Technology*, chapter Ontologies, Meta-Models and the Model-Driven Paradigm. Springer, 2006 (to appear).

[7] R. Atterer, A. Schmidt, and H. Hussmann. Extending Web Engineering Models and Tools for Automatic Usability Validation. *Journal of Web Engineering*, 2005.

[8] L. Baresi, S. Morasca, and P. Paolini. Estimating the Design Effort of Web Applications. In *Proceedings of the 9th International Software Metrics Symposium (METRICS'03)*. Springer, 2003.

[9] A. L. Baroni, S. Braz, and F. Brito e Abreu. Using OCL to Formalize Object-Oriented Design Metrics Definitions. In *Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QUAOOSE'02)*, 2002.

[10] A. L. Baroni and F. Brito e Abreu. Formalizing Object-Oriented Design Metrics upon the UML Meta-Model. In *Brazilian Symposium on Software Engineering*, 2002.

[11] L. Briand, S. Morasca, and V. R. Basili. Defining and Validating Measures for Object-Based High-Level Design. *IEEE Transactionson Software Engineering*, 25(5):722–743, 10 1999.

[12] C. Cachero and S. Meliá. The OO-H Navigation Metamodel and Profile. http://www.dlsi.ua.es/ ccachero/OOHProfile.pdf, 04 2006.

[13] C. Calero, J. Ruiz, and M. Piattini. A Web Metrics Survey Using WQM. In *Proceedings of the 4th International Conference on Web Engineering (ICWE'04)*. Springer, 2004.

[14] S. Comai, M. Matera, and A. Maurino. A Model and an XSL Framework for Analyzing the Quality of WebML Conceptual Schemas. In *Proceedings of the ER'02 International Workshop on Conceptual Modeling Quality*, pages 339 – 350. Springer, 10 2002.

[15] F. García, M.F. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, and M. Genero. Towards a consistent terminology for software measurement. *Information and Software Technology*, pages 1–14, 07 2005.

[16] M. Genero. *Defining and Validating Metrics for Conceptual Models*. PhD thesis, University of Castilla-La Mancha, 2002.

[17] J. Gómez, C. Cachero, and O. Pastor. Conceptual Modelling of Device-Independent Web Applications. *IEEE Multimedia Special Issue on Web Engineering*, 8(2):20–32, 04 2001.

[18] L. Heuser. The real world or Web Wngineering? In *Proceedings of the 4th International Conference on Web Engineering*, volume 3140, pages 1–5. Springer, 06 2004.

[19] M. Ivory. *Automated Web Site Evaluation*. Kluwer Academic Publishers, 2004.

[20] S. Kent. The Expressive Power of UML-based Engineering. In *Proceedings of the 3nd International Conference on Integrated Formal Methods*, volume 2335, page 286. Springer, 06 2002.

[21] Knowledge-based web automatic reconfigurable evaluation with guidelines optimization. http://www.isys.ucl.ac.be/bchi/research/Kwaresmi.htm.

[22] G. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. In *The Psychological Review*, volume 63, pages 81–97, 1956.

[23] J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders, 2000.

[24] Mof query/views/transformations final adopted specification. omg doc. ptc/05-11-01. www.omg.org/docs/ptc/05-11-01.pdf.

[25] L. Reynoso, M. Genero, and M. Piattini. OCL2: Using OCL in the Formal Definition of OCL Expression Measures. In *Proceedings of the 1st International Workshop on Algebraic Foundations for OCL and Applications (WAFOCA'06)*, 2006.

[26] OMG Unified Modelling Language Specification. http://www.rational.com/uml/, 06 1999.

[27] Visual web applications development environment. http://www.visualwade.com/.

[28] J. Warmer and A. Kleppe. *The Object Constraint Language. 2nd Edition. Getting your models ready for MDA*. Addison Wesley, 2003.